Fundamentos de Algoritmia

Grados en Ingeniería Informática. Grupos A, E, I

Examen Convocatoria Ordinaria, 4 de Febrero de 2021.

Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

Normas de realización del examen

1. Debes programar soluciones para cada uno de los tres ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`. Para la entrega el juez sólo tiene los datos de prueba del enunciado del problema.

2. Escribe comentarios que expliquen tu solución, justifiquen por qué se ha hecho así y ayuden a entenderla. Calcula la complejidad de todas las funciones que implementes.

3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. El nombre de usuario y contraseña que has estado utilizando durante la evaluación continua **no** son válidos.

4. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que subas al juez.

5. Tus soluciones serán evaluadas por el profesor independientemente del veredicto del juez automático. Para ello, el profesor tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.

1. (3 puntos) Dado un vector de números enteros de longitud $n > 0$, calcula el número de intervalos de una longitud dada $0 < k \leq n$, con k un valor par ($k\%2 == 0$), tales que la cantidad de valores positivos en la mitad izquierda del intervalo es mayor o igual que la cantidad de valores positivos de la mitad derecha del intervalo.

Se pide:

1. (0.75 puntos) Especificar el problema. Para ello:
   (a) Define una función lógica *contarPositivos(v,i,j)* que permita contar el número de valores positivos de un vector v en el intervalo $[i \dots j]$, siendo i y j índices del vector.
   (b) Usando la función *contarPositivos* define un predicado *hayMasPositivos(v,x,y)* que compruebe si el número de valores positivos de la mitad izquierda de un intervalo $[x \dots y]$ de un vector v es mayor o igual que el número de valores positivos de la mitad derecha, siendo x e y índices del vector.
   (c) Especifica el problema pedido utilizando el predicado anterior.
2. (1.75 puntos) Diseña e implementa un algoritmo iterativo que resuelva el problema propuesto.
3. (0.5 puntos) Escribe invariantes que permitan probar la corrección de cada uno de los bucles de tu algoritmo.
4. El problema debe resolverse con coste lineal en el número de elementos del vector.

Entrada

La entrada consta de varios casos de prueba. Cada caso de prueba tiene dos líneas, en la primera se indica el número de elementos del vector y la longitud del intervalo en que estamos interesados. En la segunda línea se indican los valores del vector. La entrada de datos termina con un valor cero.

Salida

Por cada caso de prueba el programa escribirá una línea con el número de intervalos que cumplen la propiedad pedida.

Entrada de ejemplo

```
3 2
-6 4 2
3 2
6 -4 2
3 2
6 4 -2
2 2
6 -9
2 2
-6 9
5 4
3 2 1 4 -2
5 4
3 -1 -1 2 6
6 4
2 1 -4 2 -1 -3
10 6
-1 2 -1 2 -1 2 -1 2 -1 2
10 6
1 2 3 4 5 -6 7 8 9 10
0
```

Salida de ejemplo

```
1
1
2
1
0
2
1
3
2
3
```

2.(3 puntos) Dado un vector de tamaño $n \geq 2$, de números enteros, ordenados en orden creciente, encuentra el valor mayor que falta en el vector.

Por ejemplo, dado el vector 5,8,11,14,17. El valor mayor que falta es el 16, porque es el mayor valor menor que 17 que no se encuentra en el vector.

Se pide:

1. (2 puntos) Escribe un algoritmo recursivo eficiente que permita resolver el problema para un vector dado.
2. (1 punto) Escribe la recurrencia que corresponde al coste de la función recursiva indicando claramente cual es el tamaño del problema. Haz el despliegue de la recurrencia e indica a qué orden de complejidad asintótica pertenece dicho coste.
3. No se admiten soluciones que utilicen parámetros por referencia.

Entrada

La entrada consta de varios casos de prueba. Cada caso de prueba tiene dos líneas, en la primera figura el número de elementos del vector y en la segunda sus valores. La entrada de datos termina con un valor cero.

Salida

Por cada caso de prueba el programa escribirá el valor mayor que falta en el vector.

Entrada de ejemplo

```
5
5 8 11 14 17
4
2 3 5 6
8
2 3 6 8 10 14 16 17
6
1 4 5 6 7 8
0
```

Salida de ejemplo

```
16
4
15
3
```

3. (4 puntos) En el cuento de la bella durmiente, el rey y la reina organizan un banquete para conmemorar el primer cumpleaños de la princesa Aurora. Invitan a 12 hadas buenas que le dan diversos dones a la niña, dejando fuera al hada mala, que entonces se presenta sin ser invitada. El motivo por el que este hada no es invitada a la fiesta varía de unas versiones a otras. En unas los reyes simplemente se olvidan de ella, mientras en otras el problema es que no hay suficientes platos de oro.

Vamos a realizar un programa que ayude a los reyes a seleccionar los invitados a partir de una lista de todos los habitantes del reino, de forma que el hada mala, disfrazada entre los habitantes no quede fuera. Los reyes quieren que se asigne a cada invitado un puesto en el banquete de forma que se maximice la satisfacción de los habitantes. Para ello cuentan con información sobre la satisfacción que a cada habitante del reino le produce cada uno de los puestos en el banquete. Se asignará cada puesto a aquel habitante con el que se consiga que la suma de las satisfacciones de todos los invitados sea máxima.

Tenemos las siguientes restricciones:

1. A nadie se le asignará un puesto para el que tenga satisfacción negativa. Prefiere no ir antes de utilizar ese sitio.
2. Los habitantes del reino son muy supersticiosos y creen que da mala suerte si a alguno de ellos se le asigna un puesto cuyo número coincida con el número que tiene esa persona en la lista. Por lo tanto no admitiremos ninguna solución que asigne más de un tercio de los puestos totales a invitados cuyo número en la lista sea igual al del puesto. Al calcular el tercio de los puestos se truncará el resultado.
3. El hada mala, cuyo identificador nos dan en la entrada de datos, debe estar incluida entre los invitados, no se considera necesario que ocupe ningún puesto especial.

Se pide:

- (3 puntos) Implementa un algoritmo de vuelta atrás que resuelva el problema. Explica el árbol de exploración que has utilizado y el vector en el que construyes la solución.
- (1 puntos) Plantea una estimación para podar el árbol de exploración del problema e impleméntala en tu algoritmo.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso de prueba consta de varias líneas.

En la primera se muestra el número de habitantes del reino que pueden ser invitados al convite ($n > 0$); el número de invitados que vamos a seleccionar (coincide con el número de puestos en el banquete) ($0 < m \leq n$) y el número de habitante que corresponde al hada mala que debemos invitar ($0 \leq hada < n$).

En las siguientes m líneas se da la matriz de satisfacción de cada persona con cada puesto. La fila i-ésima ($0 \leq i < m$) contiene la satisfacción de cada habitante con el puesto i-ésimo. La satisfacción que siente una persona por un puesto es un número entero. Un valor mayor indica una satisfacción mayor.

El final de datos se marca con un cero.

Salida

Para cada caso de prueba se escribe en una línea la satisfacción mejor que se puede conseguir y el número de combinaciones posibles de invitados que tenemos con esa satisfacción mayor. Si no existe solución posible se escribirá `No`.

En el primer caso, las dos combinaciones con satisfacción máxima son: 0 2 3 y 2 1 3.

En el tercer caso, la única combinación con satisfacción máxima es 2 0 1.

En el cuarto caso las tres combinaciones que tienen satisfacción máxima son: 0 2 1; 1 0 2 y 2 1 0.

Entrada de ejemplo

```
4 3 2
8 -1 3 5
-2 10 5 3
3 3 3 5
5 2 0
-1 5 5 5 5
-4 3 3 3 3
3 3 2
10 -1 2
2 10 -6
-3 2 10
3 3 2
10 2 2
2 10 2
2 2 10
0
```

Salida de ejemplo

```
18 2
No
6 1
14 3
```